Prototyping "In The Wild" Interaction Scenarios With RE/Tk

Aneesh P. Tarun
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
aneesh@ryerson.ca

Ali Mazalek
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
mazalek@ryerson.ca

Andrea Bellucci
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911, Leganés, Madrid, Spain
abellucc@inf.uc3m.es

Abstract
Building interactive environments that blend digital information into the physical world is hindered by the complexity of setting up the technological medium. We developed the Responsive Ecologies Toolkit (RE/Tk) to provide researchers and developers with a toolkit that would cut down the low-level technical demands thus making it easier to prototype applications for heterogeneous networked devices. This position paper argues for a better conceptual model to support design of interaction experiences "in the wild". It also proposes extensions to the RE/Tk to support design iterations where the potential interaction devices and their capabilities are not known.

Author Keywords
Toolkit; Prototyping; Cross-Device Interfaces; Interactive Environments.

ACM Classification Keywords
H.5.2. Information Interfaces. User Interfaces – input devices and strategies, prototyping.

Introduction
A growing body of research in cross-device interfaces has focused on providing interaction techniques for sharing information across devices [1], as well as mapping gestures across devices [3]. Much of the effort

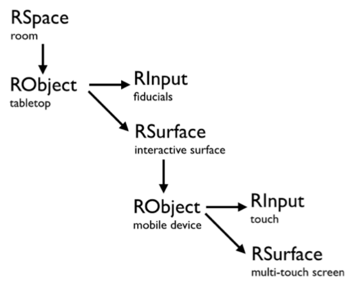


Figure 1: An example of the nested hierarchical structure.

put into such research involves bootstrapping low-level technical challenges such as programming for multiple platforms, communication across devices, and developing an ad-hoc framework for comprehending the complexity of cross-device interfaces. This has led researchers to revisit a more fundamental question in cross-device interface research. What tools and techniques are beneficial for simplifying the exploration of cross-device interaction techniques?

There exist numerous software toolkits for the rapid prototyping of spatially-aware interactions [5], tangible and physical computing [2] [4] and cross-device interactions [7][8][9]. However, there is limited support for the design of complex interactive environments involving heterogeneous off-the-shelf as well as custom devices. Weave [8], for instance, provides an authoring environment for cross-device interactions which supports off-the-shelf wearable and mobile devices. WatchCONNECT [8] explores sensor-based interactions focusing on smartwatches. Panelrama [7] aims at easing the development of distributed user interfaces, thus it does not consider smart objects without a display surface.

Our goal with RE/Tk is to provide support to quickly build responsive and interactive environments, which can include people personal devices, interactive surfaces and custom-made tangible objects (e.g., arduino-based devices). In addition, we wish to provide a fully customizable and extensible toolkit that supports multiple aspects of developing cross-device applications.

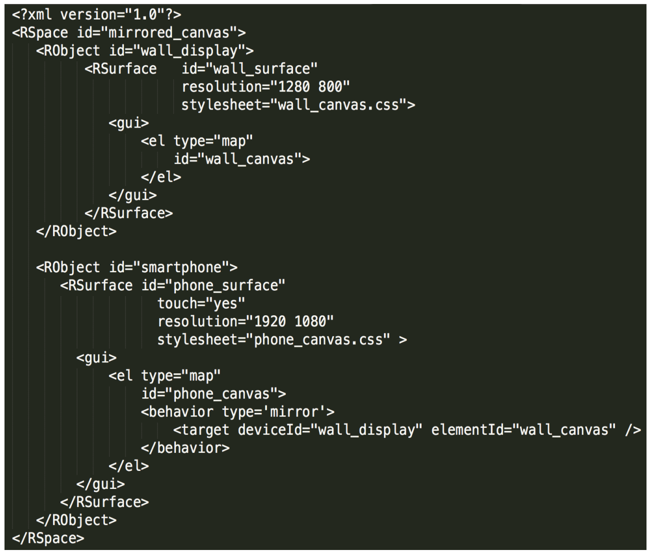
There are several reasons for this. Firstly, the landscape of interaction devices and their capabilities is ever-changing. The heterogeneity of the underlying runtime environments and the communication protocols of these devices makes it challenging to provide a toolkit that stands the test of time. In addition, scaling up or modifying existing toolkits to support cross-device interface development does not address the underlying fundamental issue: there is no conceptual framework that allows designers to visualize and discuss a complex interaction scenario at a higher level of abstraction.

We designed the Responsive Ecologies Toolkit (RE/Tk), building on the work of the ROSS Toolkit [6], for prototyping applications that span across different (off-the-shelf and custom) devices. The RE/Tk provides a conceptual framework for designing multi-device applications and toolkit for prototyping such applications.

However, in its current state, the toolkit assumes that the "players" in the interaction space i.e. the underlying devices and sensors are known before hand and conform to a fixed device hierarchy within the interaction space. Below, we discuss the current state of the RE/Tk and propose a conceptual framework and new features to the toolkit that will enable prototyping & development of "in the wild" interaction experiences such as bring-your-own-device (BYOD) scenarios.

RE/Tk

RE/Tk provides a conceptual framework that allows the developers to conceptualize and design interactive environments as hierarchical nested structures. Every object, screen, sensor in an interaction space is mapped within a hierarchical structure. This hierarchical tree (Figure 1) encapsulates relationships between various entities and determines how they interact: (a) direct interaction and communication occurs between a child node and its parent node, (b) interaction between sibling nodes is mediated by their parent node, (c) the tree structure is also used to determine the communication path between two (directly unrelated) nodes, and (d) the structure allows for easier computation of spatial interactions between different devices.



```xml
<?xml version="1.0"?>
<RSpace id="mirrored_canvas">
   <RObject id="wall_display">
         <RSurface   id="wall_surface"
                     resolution="1280 800"
                     stylesheet="wall_canvas.css">
            <gui>
                <el type="map"
                    id="wall_canvas">
                </el>
            </gui>
         </RSurface>
   </RObject>

   <RObject id="smartphone">
      <RSurface id="phone_surface"
                  touch="yes"
                  resolution="1920 1080"
                  stylesheet="phone_canvas.css" >
        <gui>
            <el type="map"
                id="phone_canvas">
                <behavior type='mirror'>
                    <target deviceId="wall_display" elementId="wall_canvas" />
                </behavior>
            </el>
        </gui>
      </RSurface>
   </RObject>
</RSpace>
```

Figure 2: An example XML descriptor file for a scenario where a wall display mirrors the interactions on a smartphone.

Designers and developers outline this structure in an XML descriptor file (see Figure 2). This document is fed to the toolkit which generates the application code and manages the communication between different sensors and devices.

All generated server and client components are in JavaScript. The server application runs on Node.js (a cross-platform runtime environment) while the client applications run on the devices' browsers. As an exception to the rule, we generate deployable code for Arduino and other microcontrollers that do not support a JavaScript runtime environment.

A Javascript API exposes the functionality of the toolkit and provides an alternative way of developing or iterating the generated interaction software for expert developers.

In addition, to support “moving targets”, RE/Tk is designed to be extensible. Designers and developers can extend the XML structure as well as the JavaScript API to suit custom workflows.

This approach simplifies the development, deployment and management of applications onto a variety of hardware and software platforms. Since the JavaScript code is compiled at runtime, modifying a part of the application code does not require recompilation for the entire interactive environment. Code can also be locally modified and inspected for debugging purposes. These features encourage faster iteration of application designs.

In addition, the toolkit provides different levels of abstractions in terms of application behavior, GUI and widgets, sensor mapping, spatial mapping and communication.

Application Behavior Abstraction. XML-based application authoring is the first level of abstraction provided by the toolkit. All the features and functions of the API are abstracted and accessible as XML tags. Large and complex XML files can also be split into multiple XML files, each defining a different feature of an application, i.e. one XML file for describing the application functionality and another one for the UI.

GUI Abstraction and Widgets. An authored XML description includes not only the device structure but also the user interface design for all the devices. This provides a unified way of designing UIs for different types of screens. This also allows for easily porting applications to native code when needed. In addition to this, we have implemented a few generic widgets (e.g. Maps widget) that can easily be included in any application.

Sensor Mapping Abstraction. RE/Tk supports straightforward mapping of sensor output data of one device as an input to a function of another device or an actuator in just a few lines. The underlying implementation of listener events, data range mapping, network routing etc. are abstracted from the developers (they are still accessible to the developers through the generated application code for modifications). Custom data filters can also be easily attached to a sensor output.

Spatial Mapping Abstraction. The hierarchical nested structure of devices is leveraged to provide easy access to spatial information of each device. Developers can directly query position information of each device relative to the interaction space or another device. Converting spatial data from one coordinate system to another is also supported by the built-in functions.

Communication Abstraction. RE/Tk uses and builds upon TUIO to simplify cross-device communication. Developers can extend the protocol for custom scenarios. The hierarchical structure, used to automatically setup the underlying communication channels between devices, forms another level of communication abstraction.

In addition to these abstractions, the modular nature of the XML files (and the API in general) allows developers to easily extend the features of RE/Tk.

Supporting interactions *in the wild*

An interaction environment with a fixed network of devices poses sufficient challenges for a developer to envision and deploy interactive experiences. However, prior access to the network information, device capabilities, and access control provides a handle to the developer for managing the overall interaction experience. RE/Tk leverages this information to simplify authoring of interactive applications. "In the wild" scenarios involve additional challenges that need to be addressed while prototyping.

In this section we discuss some novel features for the RE/Tk to support interactive BYOD scenarios. One assumption we make about the BYOD scenarios is that the developers of such scenarios have access to at least one device within the interaction environment that will be assigned the role of a server/arbiter. An example of a typical BYOD scenario: an interactive surface computer in a public space that allows walk-in users to lay their smartphones on the surface and interact with the displayed information on their smartphones as well as the larger surface.

Challenge 1: Since the interaction devices and their capabilities are not know during the design process, the toolkit has to provide a new framework and additional tools to design adaptive experiences. We propose the following two features to address this challenge.

Designing for device categories: We propose to extend the hierarchical tree model to support for hierarchies of generic device categories rather than specific devices. This allows the developers to think about and design experiences for groups of devices while the underlying toolkit manages the heterogeneous nature of such groups. Developers can declare a category of devices (e.g., smartphones or smartwatches) that interact within an interactive space for a given scenario. In addition, the developers can set the maximum number of devices supported within a device category based on the context and limitations of use. This information is used to deploy a generic Javascript application for each declared device category. We believe that this conceptual framework breaks down the barrier for entry for authoring such interaction scenarios.

Supporting adaptability of UI and interactions: Breaking down of application features by device categories may not be sufficiently granular for certain interaction scenarios. In addition, this does not guarantee that an application will run on all the *walk-in* devices as envisioned by the developer. We propose custom XML tags that allow for declaring multiple versions of the user interface or UI elements and interaction logic during the design phase. The developers can include branching logic within these tags that may include specific device properties (eg. specific UI layouts are targeted for specific range of screen sizes OR specific interaction scenarios run only if an accelerometer is present) or specific user permissions. These alternate versions of interface or interaction logic are selected & deployed at runtime. This is made possible by the automated API calls that query a newly joined interaction device for its capabilities. This approach allows the deployment of a generic interaction scenario with high level of adaptability at runtime. It also provides an easy method for developers to support graceful degradation of rich interactive experiences.

Challenge 2: BYOD scenarios may need to support authentication and permission management for different groups of users and different networks of devices.

While a single communication network and a common device permission may be sufficient for most BYOD scenarios, certain scenarios may require developers to support different permission levels for different user groups as well as support devices that may be on different networks altogether. We propose two additional features to extend the toolkit's capabilities.

Multi-Step Device Handshake: Currently every client application developed using the RE/Tk goes through a single-step handshake procedure which queries the device for its capabilities to automatically identify its role within the interaction scenario. We introduce multi-step device handshake procedure to include additional initialization routines. Firstly, we extend the device feature querying to be exhaustive that better supports the adaptability of UI and interactions previously discussed. In addition, we include an authentication phase that allows a walk-in device to be authenticated as a valid interaction device and be assigned a specific permission level. This may happen prior to the application launch as well. Lastly, we include a third step to the handshake procedure - user-side permission management. In BYOD scenarios where a walk-user can share information from her device within the interaction space, this step gives her the control to manage permissions for what gets shared. The user can also control what features of her device can be used within the interaction scenario.

Multi-step device-handshake procedures can be customized via the XML files and extended to more steps as required.

Device Authentication and Heterogeneous Networks: There are two points of entry for a typical BYOD scenario. A walk-in user can connect to a local wireless network thereby gaining access to an interaction experience through a web browser or a native application. Alternatively, users can access a specific URL on the world wide web that points to the interactive application. Since users gain such knowledge when they are in the proximity of the physical interaction space, this is a sufficient level of authentication for most BYOD scenarios.

For further permission management, web-based applications generated by the RE/Tk can take advantage of both the points of entry. Multiple access points can be placed for multiple levels of authentication to the same interactive experience. A simpler approach is to provide a unique passphrase that walk-in users can enter during the initial application launch. The passphrase, combined with the branching interaction logic previously mentioned, provides sufficient granularity for multiple levels of access control. The web-based nature of application and access control ensures that devices in different networks can be managed easily through a single interface as long as the underlying application is deployed onto the world wide web. In situations where devices are connected via Bluetooth, USB or other forms of communication, the RE/Tk provides custom modules to connect with different communication protocols. All of the different devices and protocols communicate with a server-side routing logic that is abstracted by the API. This ensures that the developer can ensure homogeneity of communication and interaction among heterogeneous mix of devices and networks with minimal effort.

Workshop Goals
Our proposed changes to the RE/Tk and the underlying conceptual framework enables us to provide a powerful prototyping tool that simplifies the design of interaction scenarios where target devices are not known. We hope to engage the community in a discussion of further opportunities and challenges for tools that support interactions "in the wild". In addition, we wish to contribute to the efforts of standardizing protocols, platforms, and tools for building cross-device interfaces.

Acknowledgements
This work has been supported by the Canada Research Chairs program and the 2015 UC3M postdoctoral mobility scholarship.

References
1. Xiang ‘Anthony’ Chen, Tovi Grossman, Daniel Wigdor, & George Fitzmaurice. 2014. Duet:

exploring joint interactions on a smart phone and a smart watch. *In Proc. CHI'14*, 159-168.

2. Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proc. UIST'01*.
3. Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: enabling and understanding cross-device interaction. In Proc. *CHI '14*, 2773-2782.
4. Scott R. Klemmer and James A. Landay. 2009. Toolkit Support for Integrating Physical and Digital Interactions. *Human-Computer Interaction* 24(3), 315–366.
5. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In Proc. *CHI'11*, 315–324.
6. Aneesh P. Tarun, Ahmed S. Arif, Andrea Bellucci, & Ali Mazalek. 2015. Responsive Objects, Surfaces and Spaces (ROSS): Framework for Simplifying Cross-Device Communication. In *TEI'15 Workshop on Interactive Infrastructures – Towards a Language for Distributed Interfaces*.
7. Jishuo Yang and Daniel J. Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*(CHI '14). ACM, New York, NY, USA, 2783-2792.
8. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3923-3932.
9. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 1247-1256.